Faculdade de Ciências da Universidade de Lisboa

Departamento de Matemática

Instituto Superior de Ciências do Trabalho e da Empresa

Departamento de Finanças

Os Efeitos do *Algorithmic Trading* na Previsibilidade dos Mercados Financeiros

Débora Cheila Abreu Rodrigues

Mestrado em Matemática Financeira

Dissertação orientada por: Professora Doutora Diana Aldea Mendes

2019

"Aqueles que passam por nós, não vão sós, não nos deixam sós. Deixam um pouco de si, levam um pouco de nós." Antoine de Saint-Exupéry

Ao meu avô, Carlos

Agradecimentos

Quero desde já agradecer à professora e orientadora, Diana Mendes, pela disponibilidade, ajuda, incentivo, paciência e por todas as críticas e sugestões dadas durante a orientação.

Neste momento tão importante na minha vida académica, não poderia de deixar de agradecer àqueles que mesmo ausentes fisicamente estiveram diariamente comigo, àqueles que nunca me deixaram desistir e me encheram de amor, carinho e força. À minha família:

Às pessoas que tornaram tudo isto possível, os meus pais, Sabino e Madalena, que me apoiaram incondicionalmente nas minhas escolhas e que lutaram sempre para que conseguisse conquistar os meus objetivos.

Ao meu namorado, Duarte, pela paciência e companheirismo ao longo destes anos. Obrigada pelas gargalhadas nos dias mais cinzentos e pelo amor e compreensão nos tempos mais difíceis.

À minha tia Idalina, que contribuiu para o meu sucesso na vida académica sempre com uma palavra de incentivo e com um pensamento positivo.

À minha parceira de vida de há 20 anos, a qual acompanhou de perto este percurso académico e que esteve desde sempre disposta em ajudar e apoiar em tudo o que precisei. Obrigada, mana.

Resumo

Nas últimas décadas, os mercados financeiros têm sofrido grandes alterações. Cada vez mais os computadores substituem os *traders* humanos. O *algorithmic trading* é um método que utiliza modelos matemáticos avançados para ajudar a tomar decisões sobre transações no mercado financeiro. Este consiste num conjunto de instruções que um computador está programado para seguir para executar uma tarefa específica (as instruções definem o prazo, a quantidade, os limites de preços e outros critérios que possam tornar a transação mais lucrativa). Com o aumento da flexibilidade e acessibilidade aos mercados, o número de investidores aumentou consideravelmente, bem como a implementação dos seus próprios algoritmos nas estratégias de negociação. Outro aspeto positivo do *algorithmic trading* é a velocidade e precisão que um humano não é capaz de atingir.

Assim, esta dissertação tem como objetivo analisar os efeitos do *algorithmic trading* na previsibilidade dos mercados financeiros. Para tal, serão utilizadas três estratégias: o *simple moving average*, o *frequency approach* e uma rede neural artificial com *backpropagation* e aprendizagem supervisionada profunda. A primeira estratégia é a mais simples de implementar baseando-se numa média aritmética. O *frequency approach* é um algoritmo que tem como base uma abordagem probabilística de contagem de frequências. Por fim, a rede neural é um dos métodos mais exigentes computacionalmente porque requer a escolha de vários parâmetros. Através destes três métodos pretende-se analisar a tendência dos movimentos ascendentes ou descendentes do índice bolsista português, PSI 20. De modo a atingir este objetivo foi utilizado o *software* Python que contém pacotes que permitem executar estes métodos (como por exemplo, scikit learn).

Pela comparação das estratégias implementadas, verificamos que a rede neural artificial utilizada é mais eficaz do que os outros dois métodos, porque capta e imita melhor os padrões da série temporal, sendo as suas estratégias mais próximas dos retornos.

Palavras-chave: *Algorithmic Trading*, *Moving Average*, *Frequency Approach*, Redes Neurais Artificiais

Abstract

In last decades, financial markets have undergone major changes. More and more computers replace human traders. Algorithmic trading is a method that uses advanced mathematical models to help make decisions about financial market transactions. This is a set of instructions that a computer is programmed to follow in order to perform a specific task (instructions define the timeframe, quantity, price limits, and other criteria that might make the transaction more profitable). With the increasing of flexibility and market accessibility, the number of investors has increased significantly, as well as the implementation of their algorithms in trading strategies. A positive view of algorithmic trading is that's it has the speed and accuracy that a human cannot achieve.

The aim of this work is to analyse the effects of algorithmic trading on the predictability of financial markets. To this end, three strategies will be used: simple moving average, frequency approach and an artificial backpropagation neural network with deep supervised learning. The first strategy is the simplest to implement based on an arithmetic mean. Frequency approach transforms the two real-valued features to binary ones and assess the probability of an upward and a downward movement. Artificial neural network is one of most computationally demanding methods because it requires the choice and adjustments of several parameters. Through these three methods we intend to analyse the trend of an upward or downward movement of Portuguese Stock Index, PSI 20. In order to achieve this goal, we used Python software that contains packages that allow us to execute these strategies (such as scikit learn).

In fact, it is possible to compare these strategies. Thus, we find that the artificial neural network used is more effective than the other two methods because our strategy is nearest to returns, which doesn't happen in the other methods.

Keywords: Algorithmic Trading, Moving Average, Frequency Approach, Artificial Neural Networks

Índice

Lista de figuras

Lista de tabelas

Lista de abreviaturas

API	Interface de Programação de Aplicações (*Application Programming Interface*)
CAGR	Taxa de Crescimento Anual Composta (*Compound Annual Growth Rate*)
CBT	Negociação Baseada no Computador (*Computer Based Trading*)
DL	Aprendizagem Profunda (*Deep Learning*)
DNN	Rede Neural Profunda (*Deep Neural Network*)
DOT	*Designated Order Turnaround*
ECN	Rede de Comunicação Eletrónica (*Electronic Comunications Networks*)
HTF	Negociação de Alta Frequência (*High Frequency Trading*)
LBFGS	*Limited-memory* Broyden Fletcher Goldfarb Shanno
ML	Aprendizagem de Máquina (*Machine Learning*)
MLP	*Multi-Layer Perceptron*
NASDAQ	*Nacional Association of Securities Dealers Automated Quotations*
PSI 20	Índice Bolsista Português (*Portuguese Stock Index*)
ReLU	Unidade Linear Retificada (*Retified Linear Units*)
SEC	*United States Securities and Exchange Comission*

Introdução

> *"A desmaterialização dos processos no sector financeiro permitiu a substituição progressiva dos corretores de carne e osso pelos corretores automáticos invisíveis."*
>
> Albert J. Menkveld, Escola de Finanças Duisenberg em Amesterdão

Os mercados financeiros têm sido alvo de grandes mudanças. A tecnologia está a tornar-se cada vez mais um fator influenciador na vida das pessoas e os seus desenvolvimentos tecnológicos estão a gerar um impacto positivo nos mercados financeiros. Devido ao rápido desenvolvimento computacional, as transações têm-se tornado mais automáticas através do uso de plataformas de *trading* eletrónicas, permitindo o acesso de outros indivíduos aos mercados financeiros visto que antes o acesso era exclusivo a especialistas de *trading* das empresas. Consequentemente, houve um desenvolvimento dos indicadores técnicos através do processo de evolução dos algoritmos, o que permitiu que os parâmetros das estratégias fossem otimizados. O *algoritmic trading* veio facilitar o trabalho humano na tomada de decisões mais lógicas e na procura dos melhores investimentos com base na análise do mercado, calculando a probabilidade de a transação ser rentável ou não. Segundo Neil Jonhson do Departamento de Física da Universidade de Miami, o que defrontamos hoje é "um sistema de máquinas em que vemos o declínio da capacidade humana para influenciar os movimentos de preços em escalas de tempo cada vez mais pequenas."

O *algorithmic trading* ganhou importância nos mercados globais na última década, o que corresponde a quase 70% do volume total de transações nos mercados desenvolvidos. Uma das principais vantagens do *algorithmic trading* sobre o comércio comum é a rapidez, o aumento de liquidez, remoção de emoções e diminuição de custos. Um relatório recente da Thomson Reuters estima que os sistemas de negociação algorítmica agora são responsáveis por 75% do volume global de negociação. O mercado algorítmico global deverá crescer 10,2% CAGR entre 2018 e 2026. A nível mundial facilmente se verifica que a América do Norte é onde se utiliza mais o *algoritmic trading* nas transações, enquanto que na América do Sul a sua utilização é bem inferior. Podemos observar estes dados na figura seguinte:

Figura 1. Receitas do mercado global do *algorithmic trading*

O principal objetivo deste trabalho é utilizar indicadores de análise técnica com o propósito de prever a direção do preço do PSI 20 tentando maximizar o retorno obtido pela estratégia de negociação. Para tal, serão utilizados três métodos: *moving average*, *frequency approach* e redes neurais artificiais. O primeiro método a ser utilizado foi o *simple moving average* que consiste apenas na média aritmética, sendo atribuído o mesmo peso a todas as observações, mas escolhendo de quantas em quantas observações deve ser feito o alisamento. O segundo método é através de uma abordagem frequencista no qual também se utilizam números binários. O terceiro método faz parte do *Machine Learning*, o qual abrange diversas áreas científicas e tem várias versões, mas, neste caso, será utilizado uma rede neural artificial com *backpropagation* e aprendizagem supervisionada profunda. O método de otimização que iremos usar nesta rede será o LBFGS (*Limited-memory* Broyden–Fletcher–Goldfarb–Shanno) que é um método iterativo para resolver problema de otimização não linear. A função de ativação é a ReLU (*Rectified Linear Unit*), uma função não linear muito utilizada nas redes neurais com aprendizagem profundas. Os três métodos serão implementados e executados no software Python.

Quanto à estrutura do trabalho, será basicamente dividido em três partes: enquadramento teórico, metodologia e resultados e análise. No primeiro capítulo é abordado o desenvolvimento tecnológico das plataformas de negociação, os tipos de automatização dos mercados financeiros, definições de alguns conceitos importantes: liquidez, volatilidade e tendência, o conceito de *algorithmic trading* bem como as suas vantagens e desvantagens. No segundo capítulo fala-se inicialmente do índice em estudo (PSI 20) e o conceito, funcionamento e descrição da metodologia selecionada para implementação dos três métodos de *algorithmic trading*. Por fim, no terceiro capítulo são apresentados os resultados e faz-se uma análise dos mesmos.

1. Enquadramento teórico

Neste capítulo será apresentada uma visão geral da evolução dos mercados financeiros bem como os conceitos principais ao desenvolvimento deste trabalho.

1.1. Evolução da automatização dos mercados financeiros

Inicialmente, antes da utilização dos computadores, as transações eram feitas no *Floor Trading*, no qual os *traders* compravam e vendiam títulos através do método de negociação aberta (*Open Outcry*). Primeiro, um comerciante anunciava o preço pelo qual estava disposto a comprar ou vender e o número de títulos que estava disposto a negociar a esse preço; depois, os outros comerciantes, gritavam de volta caso estivessem interessados em aceitar o negócio, e se houvesse um interesse mútuo, ambos os comerciantes anotavam nos seus cadernos. No final do dia, os comerciantes liquidavam os seus cadernos para garantir que todas as suas transações fossem corretamente combinadas com uma contraparte.

Os títulos financeiros circulavam apenas em papel nos mercados financeiros, sendo definidos essencialmente como "objetos tangíveis em forma de papel que incorporam direitos de propriedade ou de dívida a quem os detém" (Nascimento, 2017). Os títulos físicos eram a prova da subscrição de produtos financeiros. Contudo, este método tem alguns pontos negativos como o facto de serem dispendiosos, vulneráveis, um grande obstáculo à transação, facilidade de perda e roubo. Um bom exemplo disso foi o ataque terrorista a 11 de setembro de 2001 no qual foram destruídos e perdidos títulos no valor de 16 mil milhões de dólares.

Em 1949, Alfred Winslow Jones usou um algoritmo para fazer um balanço entes as posições longas e curtas (*long and short position*) num *hedge fund*.

Já em 1969, com base na tecnologia de cotação por *streaming* baseada em troca digital, foi fundada a primeira plataforma de negociação eletrónica, denominada Instinet. Esta plataforma operava apenas no mercado norte americano e permitiu aos seus utilizadores transacionar ações fora das bolsas tradicionais. (Gilberto, 2015). A introdução deste sistema de negociação automatizada marcou o nascimento das transações eletrónicas e uma divergência da *Open Outcry* (método de comunicação verbal e manual usado pelos negociadores nas bolsas de valores e de futuros).

Dois anos depois, em 1971 nasceu a primeira bolsa eletrónica do mundo, a NASDAQ (*Nacional Association of Securities Dealers Automated Quotations*), plataforma computorizada de compra e venda automatizada de títulos financeiros.

Em 1977 foi implementado o sistema DOT (*Designated Order Turnaround*) na bolsa de Nova Iorque e mais tarde, em 1984, o sistema Super-DOT. O sistema DOT caracteriza-se por ser um sistema informatizado de entrada de pedidos que permite ordens de compra e venda de muitas ações para serem transmitidas imediatamente ao especialista na bolsa, onde a execução ocorrerá rapidamente. Estes sistemas tiveram um papel crucial para os mercados financeiros, uma vez que permitiram a realização de ordens de compra e venda de ações de forma eletrónica. As operações podiam ser visualizadas num equipamento denominado *display book*, que permitia a cada empresa especializada executar ordens para o mercado. Com isto, nasceu a era do *Computer Based Trading* (CBT), caracterizada pela utilização de plataformas eletrónicas para execução de transações financeiras, mas também, pela possibilidade de, de

forma automática, fazer depender as decisões de compra ou venda da verificação de valores previamente inseridos no sistema. (Sousa, 2016)

Na década de 1980, a bolsa de Nova Iorque já funcionava de forma totalmente eletrónica, contudo, estes processos ainda dependiam muito da componente humana. Apesar de o *Computer Based Trading* assentar em operações relativamente simples, a 19 de outubro de 1987 deu-se o *crash* conhecido como "segunda feira negra", no qual o índice norte-americano Dow Jones registou uma perda de 22,6%, sendo o CBT o principal potenciador da situação. Contrariamente ao que era esperado, a utilização da computação nos mercados financeiros evoluiu significativamente.

No fim da década de 90, houve um desenvolvimento da negociação eletrónica através do um novo sistema eletrónico de transações financeiras, chamado *Electronic Comunications Networks* (ECN) que permitiu a todos os investidores a realização de transações financeiras fora dos mercados tradicionais, a qualquer hora e com custos reduzidos de transação.

A gradual eficiência da negociação eletrónica, aliada à sua rapidez, reduzidos custos e pequena margem de erro, conduziu a um aumento do investimento na tecnologia e nos recursos necessários à sua execução (Oliveira, 2017). Consequentemente, surgiu a *Regulation Alternative Trading Systems* - plataformas eletrónicas como meio alternativo de efetuar transações financeiras- aprovada em 1998 pela SEC (*U.S. Securities and Exchange Comission*).

Nesta última década, é possível verificar a crescente capacidade de automatização e processamento dos sistemas de negociação, devido à inovação e evolução da tecnologia informática, o que tornou os sistemas de negociação mais eficientes e com baixa liquidez. A customização dos algoritmos representa a maior área de crescimento no universo do *algorithmic trading*.

De acordo com o *Global Algorithmic Trading Market 2018-2022 report*, publicado pela *Research and Markets*, espera-se que o mercado global de *algorithmic trading* cresça a um *CAGR (Compound Annual Growth Rate)* de 10,36% durante o período 2018-2022.

1.2. Tipos de automatização dos mercados financeiros

***Algorithmic Trading*:** É o processo de converter uma estratégia de negociação em um algoritmo ou código de computador e executá-las. O *algorithmic trading* permite que um *trader* defina instruções para um computador comprar ou vender títulos financeiros quando as condições definidas forem satisfeitas.

***Quantitative Trading*:** A negociação quantitativa envolve o uso de cálculos matemáticos e estatísticos avançados, juntamente com a análise quantitativa, para elaborar estratégias de negociação que possam ser executadas manualmente ou de forma automatizada.

Automated Trading: A negociação automatizada é a automatização absoluta do processo de negociação. As decisões de compra e venda são tomadas por programas de computador, ou seja, o pedido é criado automaticamente, enviado para o mercado e executado.

High Frequency Trading (HFT): Este tipo de negociação executa ordens em um período de tempo extremamente curto, normalmente em nanossegundos, atingindo um lucro muito pequeno de cada transação, mas fazendo um grande volume de transações.

1.3. Liquidez

A qualidade do mercado é determinada principalmente pela liquidez. Por liquidez entende-se a facilidade de um ativo ser transformado em dinheiro sem perdas significativas no seu valor, em outras palavras, refere-se à agilidade com que um investidor consegue se desfazer de um investimento para voltar a ter dinheiro sem que, para isso, precise de ter prejuízo. O *algorithmic trading* produz um efeito positivo no mercado, tornando-o mais líquido. *Algorithmic traders* consomem liquidez quando é barato e fornecem liquidez quando é caro. No geral, o *algorithmic trading* monitoriza de perto o mercado em termos de liquidez e informações e reage rapidamente a mudanças nas condições do mercado, fornecendo liquidez em situações de stress do mercado.

1.4. Volatilidade

A volatilidade é designada como a sensibilidade de um determinado ativo às variações dos mercados financeiros. Uma alta variabilidade nos preços dos ativos indica uma grande incerteza sobre o valor do ativo subjacente, potenciando decisões incorretas de investimento quando a variabilidade dos preços é alta. A variabilidade do preço decresce quando se simula estratégias com base no *algorithmic trading*. Os *algorithmic traders* fornecem liquidez mesmo que os mercados se tornem turbulentos. Assim, os algoritmos atenuam as flutuações de preço e contribuem para a robustez dos mercados em momentos de stress. É possível medir a taxa de variação de um ativo num determinado período de tempo utilizando o Índice de volatilidade.

1.5. Tendência

Um dos conceitos mais importantes da análise técnica é o de tendência. De facto, é através da observação das tendências mais acentuadas do mercado, e antecipando as futuras tendências, que é possível programar estratégias de investimento coerentes e, portanto, rentáveis.

É possível descrever o mercado como *bull* e *bear* que descrevem uma tendência de mercado ascendente e descendente, respetivamente.

As tendências em relação à extensão designam-se por tendência a curto prazo no qual não dura mais do que três semanas; tendência a médio prazo com duração em três semanas e vários meses e a tendência de longo prazo que dura pelo menos um ano. A tendência para subir é intitulada como ascendente e a tendência para descer como descendente e se não subir nem descer é chamada estacionária.

1.6. Conceito: *Algorithmic Trading*

O *algorithmic trading* é a transação de títulos financeiros com base na decisão de compra ou venda através de métodos computacionais. Os algoritmos são, muitas vezes, programados pelos *traders* tendo em conta a estratégia a utilizar.

A negociação por meio de algoritmos exige que os investidores especifiquem primeiro as suas metas de investimento e/ou negociação em termos de instruções matemáticas. Dependendo das necessidades dos investidores, as instruções personalizadas variam de simples a altamente sofisticadas. Depois de as instruções serem especificadas, os computadores implementam essas negociações seguindo as instruções prescritas. Os fundos de gestão de dinheiro, fundos mútuos e índices, planos de pensão, fundos quantitativos e até fundos de *hedge* usam algoritmos para implementar decisões de investimento. Os algoritmos determinam o preço apropriado, tempo e quantidade de ações para entrar no mercado.

Indiscutivelmente, existem algoritmos mais eficazes do que outros, consoante, por exemplo, a quantidade de informação a que conseguem aceder para que a ordem que emitem seja a mais segura e eficiente. No entanto, revela-se essencial assegurar que a complexidade dos algoritmos criados não põe em causa a velocidade de realização das transações.

1.6.1. Vantagens do *Algorithmic Trading*

Atualmente, o *algorithmic trading* tornou-se muito popular devido às vantagens que detém em relação à negociação manual. As vantagens do *algorithmic trading* estão relacionadas com a velocidade, precisão e custos reduzidos.

A velocidade de execução das transações aumentou significativamente devido ao uso dos computadores e tecnologia para implementar tais negociações. Os algoritmos tomam decisões em microssegundos e executam facilmente múltiplos negócios simultaneamente, algo que os seres humanos não conseguem fazer porque não têm capacidade de reagir tão rápido quanto uma máquina. O *algorithmic trading* consegue monitorizar e analisar um número muito maior de títulos financeiros, em comparação com um indivíduo a fazê-lo manualmente. O facto de as negociações poderem ser analisadas e executadas mais rapidamente permite que surjam mais oportunidades a melhores preços.

As emoções humanas podem levar os investidores a tomar decisões irracionais com base no medo ou na ganância. Portanto, a remoção de emoções humanas é uma grande vantagem uma vez que as negociações são efetuadas tendo por base um conjunto de critérios e condições pré-definidas, reduzindo a probabilidade de cometer erros humanos.

À medida que o processo é maioritariamente executado pela máquina, é reduzida a quantidade de erros humanos causados pela fadiga causada durante a execução de um grande número de pedidos manualmente. A automatização dos mercados financeiros é mais precisa do que a inserção dos dados manualmente, uma vez que reduz substancialmente a taxa de erro. Os operadores não precisam de intervir sempre no mercado visto que os negócios podem ser executados sem supervisão contínua, reduzindo assim os custos de transação.

Uma mais valia é o *backtest*, no qual o *trader* simula uma estratégia de *trading* com base nos dados históricos para analisar o risco e a previsão antes de transacionar qualquer título financeiro.

1.6.2. Desvantagens do *Algorithmic Trading*

Apesar dos inúmeros benefícios do *algorithmic trading*, existem também algumas desvantagens. As novas tecnologias não só representam oportunidades, mas também riscos. Estes estão geralmente associados a investimentos na gestão de riscos existentes como por exemplo, nos riscos associados às falhas de sistema, conectividade de rede ou problemas de desempenho.

Os algoritmos são feitos por humanos e, portanto, há uma probabilidade de haver um engano. Assim, outros riscos estão associados aos algoritmos que contêm erros ou criam um efeito cascata (quando um acontecimento repercute em diversos outros que se sucedem sem parar) e aumentam a volatilidade dos preços.

Muitas vezes, o elevado número de entradas de pedidos, alterações ou exclusões dentro de um período muito curto podem levar à sobrecarga do sistema de negociação.

2. Metodologia

O investimento em títulos financeiros requer um certo conhecimento em termos de análise dos mercados. Para análise técnica financeira, existem indicadores muito populares em função da sua confiabilidade e grau de complexidade.

Neste trabalho serão avaliadas as estratégias de *algorithmic trading*, mais especificamente, estratégias direcionais de negociação. Essas estratégias precisam de algum poder de previsão para ter sucesso nas suas operações. Neste contexto, será explorada a capacidade de predição de diferentes técnicas abordadas para previsão de tendências de subida ou descida nos preços dos títulos financeiros. Caso seja identificada uma tendência, a estratégia de negociação inicia uma operação de compra e venda de títulos. Com isso, espera-se melhorar a eficiência das negociações realizadas a fim de aumentar o lucro e reduzir o risco em suas operações.

Assim, serão aplicadas três estratégias de *algorithmic trading* a dados de uma série temporal financeira: *simple moving average* (SMA), *frequency approach* e redes neurais profundas (DNN), a fim de classificar as direções do movimento do mercado financeiro. A escolha destes três métodos não foi ao acaso. Foram escolhidos três indicadores com graus de complexidade diferentes, o SMA é indicador simples e um dos mais usados, enquanto que as DNN são métodos muito mais recentes, mais complexos e com resultados significativos em várias áreas.

Neste trabalho utilizaremos o PSI 20, o principal índice de referência do mercado de capitais português. Os dados a ser considerados são os valores de fecho, com dados diários (5 vezes por semana). O período escolhido é de 04/01/2010 a 03/07/2019. O objetivo deste estudo é avaliar os efeitos do *algorithmic trading* na previsibilidade dos mercados financeiros tendo em contas as diferentes técnicas acima mencionadas.

A Figura 2 ilustra de forma esquemática o fluxo num processo de *algorithmic trading*. Como se pode observar, são necessários vários passos e etapas de pré-processamentos e análise dos dados até quando um algoritmo está pronto para *trading*, incluindo configurações, execuções experimentais e simulações de cenários e riscos possíveis.

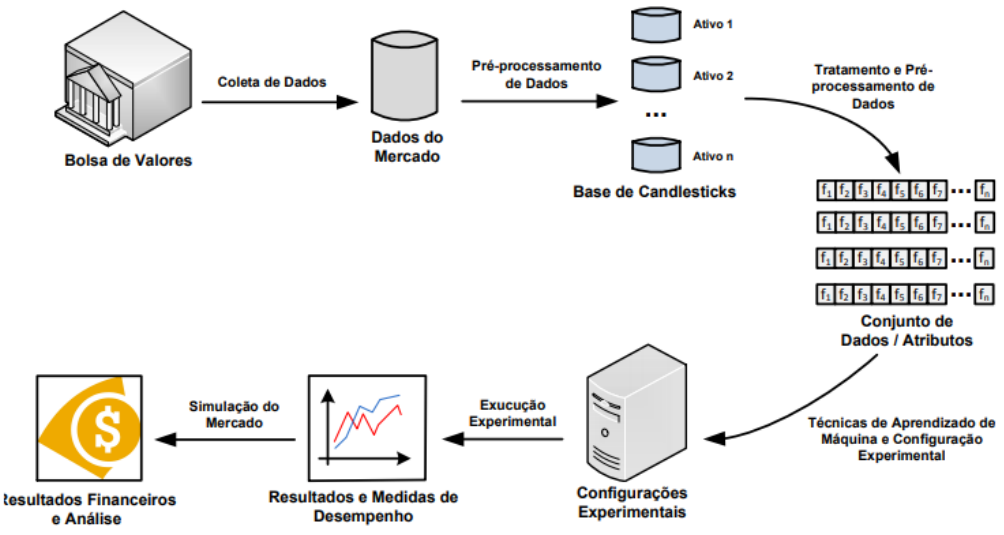

Figura 2. Processamento do *algorithmic trading*

2.1. PSI 20

PSI 20, também conhecido como Portuguese Stock Index, é o principal índice da Euronext Lisboa. Este é um indicador da evolução do mercado de ações em Portugal e reflete a evolução dos preços das 20 ações emitidas com maior dimensão e liquidez selecionadas no conjunto das empresas cotadas na Euronext Lisboa. O PSI 20 foi lançado com duas principais funções: uma delas é servir de indicador da evolução do mercado acionista português e servir de suporte à negociação de contratos de futuros e opções. O valor base do PSI 20 remonta a 31 de dezembro de 1992 e foi de 3000 pontos.

A capitalização bolsista das emissões que compõem o PSI 20 é ajustada pelo *free float*, não podendo cada emissão ter uma ponderação superior a 20% nas datas de revisão periódica da carteira. Devido às suas características, o índice PSI 20 foi selecionado pelo mercado para servir de subjacente a produtos estruturados, cuja rentabilidade depende, de uma ou de outra forma, do comportamento do mercado bolsista português.

Figura 3. Dados históricos do PSI 20 no período em estudo

2.2. Métodos do *Algorithmic Trading*

Apesar de o computador iniciar as negociações, o utilizador ainda tem a capacidade de inserir a estratégia que deseja usar, podendo decidir o volume, o preço e a que horas deverá acontecer a transação. Assim, as estratégias utilizadas pelos investidores podem ter um grande impacto nos seus ganhos.

Uma estratégia de *algorithmic trading* consiste na utilização de dados de mercado (históricos ou ativos) num programa de computador. Em seguida, o programa envia pedidos ao broker através de uma API (*Application Programming Interface*) e recebe de volta notificações do status do pedido do *broker*. O fluxograma na figura 4 ilustra este processo.

Qualquer estratégia de *trading* tem de incluir informações específicas para quando a negociação é aberta, e também um sistema de gestão de risco para evitar perdas. Isto aumenta a probabilidade geral de negociar com sucesso a longo-prazo.

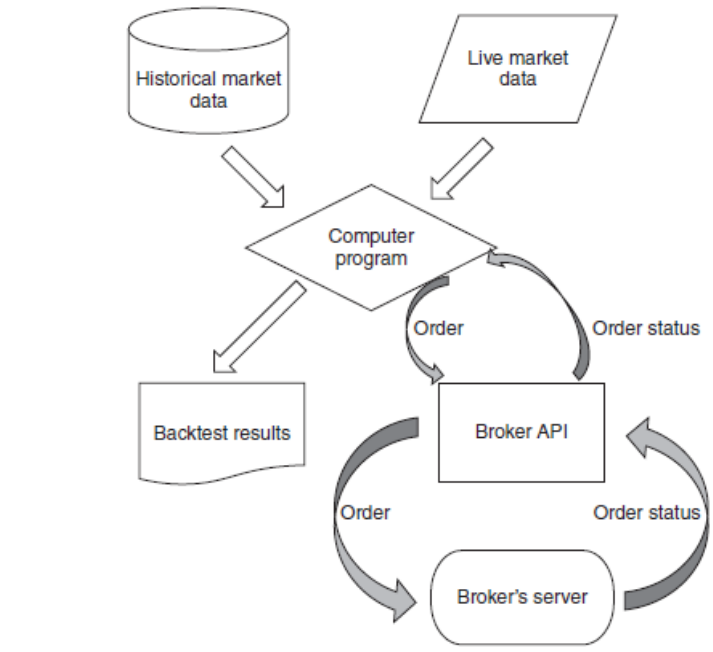

Figura 4. Funcionamento do *algorithmic trading*

Normalmente, as estratégias mais comuns usadas no *algorithmic trading* são baseadas em tendência (o utilizador segue as tendências do mercado usando os dados históricos e transaciona com base nisso), estratégia de reversão de média (baseia-se no princípio de que, embora os preços subam ou desçam, é apenas temporário e, eventualmente, retornam ao seu preço médio, assim o programa identifica o limite superior e inferior do título financeiro e realiza negociações quando o preço fica acima ou abaixo desse intervalo). Além das anteriores, estas também podem ser baseadas nas estratégias de arbitragem (na presença de uma diferença no preço dos títulos em duas bolsas de valores, verifica-se uma oportunidade de arbitragem no mercado, que pode resultar num lucro livre de risco).

2.2.1. Simple Moving Average

O *moving average* é um indicador técnico muito popular que segue tendências com base nos preços históricos. São médias que seguem a tendência, ou seja, sobem e descem em linha com o mercado. As médias móveis ajudam a suavizar os movimentos do preço, isto é, retiram os ruídos representados pelas oscilações mais fortes, tornando-se mais simples de entender o comportamento do preço de um ativo e de identificar tendências. Além disso, esta estratégia também é utilizada para identificar a entrada na negociação.

Comprar e vender com base no *moving average* pode ser uma estratégia eficaz para administrar o risco de grandes perdas dos principais mercados em baixa. Quando o preço de fecho do mercado está acima do valor do *moving average*, mantém-se o índice, e caso o preço termine abaixo, passamo-lo para dinheiro. A desvantagem desta estratégia é o facto de não proporcionar grandes lucros, porém, também traz grandes perdas. Além disso, ela pode produzir o ocasional whipsaw (sinal de compra ou venda de curto prazo). A aplicação do *moving average* no gráfico, fornece um sinal de compra quando o gráfico ultrapassa o *moving average* com o corpo da vela. Por outro lado, é fornecido um sinal de venda quando o gráfico desce abaixo do *moving average* com o corpo da vela. No caso de serem dois *moving average* para dois períodos diferentes, é fornecido um sinal de compra quando o *moving average* de menor período ultrapassa o *moving average* do período de tempo mais longo. Por outro lado, é fornecido um sinal de venda quando o *moving average* de menor prazo fica abaixo do *moving average* de maior prazo. (Mislinski, 2019)

A direção do *moving average* mostra a direção dos movimentos dos preços. Quanto menor o período do *moving average*, mais sinais falsos podem ocorrer, e quanto maior for o período, maior é o atraso do indicador. Para aumentar a sensibilidade do *moving average*, é preciso diminuir o seu período. O uso das médias móveis é mais confiável quando a tendência está evidente.

A maioria dos *moving average* calculam-se com base nos preços de fecho. Uma das técnicas de análise mais populares é o *simple moving average* que é calculado somando os preços de fecho de um mercado para um número de períodos escolhidos e dividindo esse número pelo número de períodos. Seja SMA_t o valor do *simple moving average* no tempo *t* e *n* o número de observações, tem-se:

$$SMA_t = \frac{1}{n} \sum_{j=t-n+1}^{t} x_j \quad \forall\, t \geq n \tag{1}$$

O *simple moving average* não acomoda influências sazonais, de tendência ou de ciclo de negócios. Este método apenas calcula a média e cada vez que esta é calculada, os dados mais antigos são descartados e os dados mais recentes são incluídos. Contudo, este método tem algumas limitações. Se houver uma alteração significativa no padrão, ele reage lentamente. Isto acontece porque atribuímos o mesmo peso a todos os períodos. Uma alternativa a este modelo seria o *weighted moving average* no qual é atribuído um peso maior ou menor consoante a importância de cada período. Em geral, se os dados forem estáveis, utilizamos o *simple moving average* com um número de períodos grande, no caso

de os dados não serem estáveis (se houver muitas oscilações dos dados) é aconselhado utilizar o *weighted moving average* e valores pequenos de *n*. Uma fraqueza desse método é que esquece rapidamente o passado. Uma vantagem é que é rápido e fácil de usar.

2.2.2. Frequency Approach

Além de algoritmos e técnicas mais sofisticados, pode-se implementar uma abordagem frequencista (*frequency approach*) para prever movimentos direcionais nos mercados financeiros. Para este fim, pode-se transformar as duas características de valor real em binárias e avaliar a probabilidade de um movimento ascendente e descendente, respetivamente, através das observações históricas de tais movimentos, dadas as quatro combinações possíveis para as duas características binárias, isto é: (0, 0), (0, 1), (1, 0), (1, 1).

2.2.3. Rede Neural Profunda

Antes de introduzirmos as Redes Neurais Artificiais, interessa-nos, primeiro, saber um pouco sobre o funcionamento do cérebro humano, mais especificamente o modo como funcionam os neurónios.

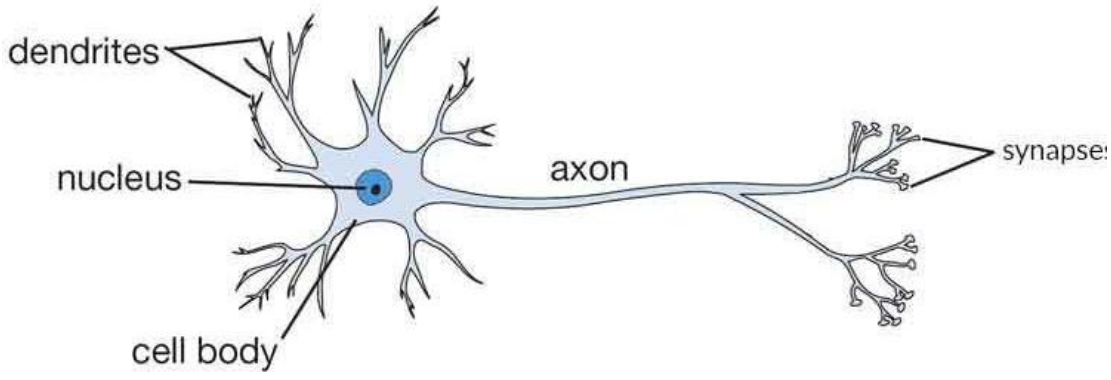

Figura 5. Estrutura de um neurónio

O cérebro humano é composto por biliões de neurónios que estão interligados, formando as redes neurais, e que são responsáveis por conduzir os impulsos elétricos para fazer a transmissão dessas informações. Os neurónios são compostos por três partes: os dendritos, que recebem informação dos outros neurónios, o corpo da célula (*cell body*) soma todos os sinais de entrada para gerar input sendo

responsável pelo processamento das informações, e o axónio, que distribui a informação. As sinapses são as junções entre a terminação de um neurónio e a membrana de outro neurónio que fazem a conexão entre células, dando continuidade à propagação do impulso nervoso por toda a rede neural. Os impulsos recebidos por um neurónio A, em um determinado momento, são processados, e atingindo um dado início de ação, o neurónio A dispara, produzindo uma substância que flui pelo axónio que está conectado a um dendrito de um outro neurónio B. À medida que um sinal passa pelo neurónio, ele pode ser amplificado (conexões excitatórias) ou atenuado (conexões inibitórias), levando em consideração a memória sobre aquela informação que está a ser processada. Como cada região do cérebro é responsável por funções específicas, essa informação percorre as redes neurais até chegar ao campo em que terá a "resposta" para aquele estímulo, a partir dos pesos atribuídos àquele sinal. Esses valores são dados levando em consideração a memorização. Sendo assim, faz sentido dizer que quanto mais treino o cérebro receber ao longo de sua vida útil, mais rápida será a resposta ao estímulo recebido. Quanto mais células trabalharem em conjunto, mais elas podem processar e mais eficaz torna-se o trabalho. Logo, para obter um melhor rendimento do sistema são necessários muitos neurónios. Este processo depende de vários fatores, como a geometria da sinapse e o tipo de neurotransmissor.

Mas afinal o que são as redes neurais artificiais? Uma rede neural artificial é uma tecnologia criada para simular a atividade do cérebro humano. O objetivo dos especialistas e pesquisadores da computação cognitiva sempre esteve centrado em replicar e superar em máquina o funcionamento do cérebro humano.

Uma Rede Neural Artificial é constituída por várias camadas: uma camada de entrada, uma de saída e zero ou mais camadas ocultas. A camada de entrada consiste em um nodo para cada variável independente, enquanto a camada de saída consiste em um ou mais nodos que correspondem à decisão final. As camadas ocultas ficam no meio, e cada uma consiste em vários nodos que recebem inputs dos nodos da camada abaixo deles e alimentam os seus outputs para os nodos das camadas acima. Uma rede neural é um conjunto de unidades de input conectadas (também chamados de neurónios, nodos ou variáveis) $X_0, X_1, X_2, \ldots, X_n$ e uma ou mais variáveis de output $Y_0, Y_1, Y_2, \ldots, Y_n$ onde cada conexão que está associada aos parâmetros indica a força dessa conexão (chamado de peso) $W_0, W_1, W_2, \ldots, Wn$. Durante a fase de aprendizagem, a rede aprende ajustando os pesos para poder prever ou classificar corretamente o output de um conjunto de amostras de inputs. Na modelagem das redes neurais artificiais, um dos inputs $X_i,\ i \in \{0, \ldots, n\}$, conhecido como *bias*, tem de ser incluído e normalmente assume o valor 1 ou -1.

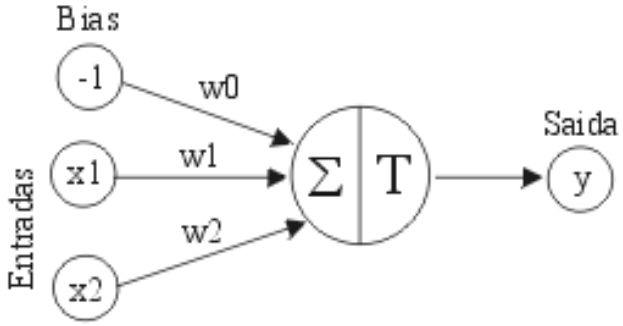

Figura 6. Rede Neural Artificial

Cada input é multiplicado pelos pesos correspondentes. Os inputs ponderados são resumidos dentro da unidade de computação (neurónio artificial). À soma ponderada soma-se o *bias*, uma constante. A expressão seguinte

$$\sum_{i=0}^{+\infty} X_i \cdot W_i + b \tag{2}$$

representa o somatório ponderado entre as sinapses de cada neurónio. A soma corresponde a qualquer valor numérico que varia entre zero e infinito. Para limitar a resposta ao valor desejado, o valor limite é configurado. Para isso, a soma é passada através de uma função de ativação. A função de ativação está definida para a função de transferência usada para obter a saída desejada. Existem funções de ativação linear e não linear. Algumas das funções de ativação mais usadas são: Sigmoide**,** Tangente Hiperbólica (TanH)**,** Unidade Linear Retificada (ReLU)**,** Unidade Linear Exponencial (ELU) e Unidade Linear Retificada com Vazamento**.**

Função de ativação ReLU

A função de ativação que será utilizada é a ReLU (unidade linear retificada), uma função não linear muito conhecida pelo seu bom desempenho computacional.

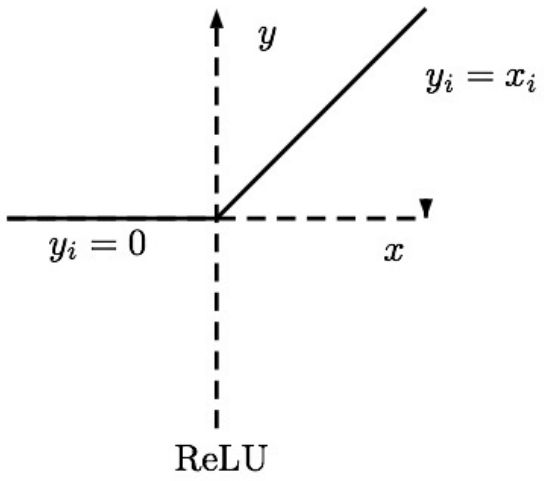

Figura 7. Representação gráfica da função de ativação ReLU

Formalmente, a ReLU é definida como:

$$f(x) = \max(0, x) = \begin{cases} 0 \text{ se } x \leq 0 \\ x \text{ se } x > 0 \end{cases} \quad (3)$$

Essa função tem propriedades muito interessantes, como ser parcialmente linear, o que facilita na hora do treino, e ter derivadas muito simples:

$$f'(x) = \begin{cases} 0 \text{ se } x < 0 \\ 1 \text{ se } x > 0 \end{cases} \quad (4)$$

Na prática, o ponto onde a derivada não está definida é implementado como se fizesse parte de alguma das regiões onde ela é bem-comportada. Quando é introduzida a não linearidade, a rede neural artificial consegue representar qualquer função, dado um número suficiente de neurónios. Quanto maior o número de neurónios, maior a capacidade do modelo. É importante realçar também que, quando se introduz a não linearidade na rede neural, a função custo a ser otimizada torna-se não convexa e extremamente complicada de otimizar, dificultando consideravelmente o processo de treino.

As Redes Neurais Artificiais (RNA) são consideradas umas das técnicas de *Machine Learning* mais eficientes da atualidade, sendo que grandes empresas têm utilizado essas técnicas em vários tipos de aplicações. Os carros que se tornam autónomos na condução, as aplicações que conseguem gerar novas músicas, escrever poemas, classificar imagens, são aplicações que foram maioritariamente construídas utilizando as redes neurais. Há algum tempo houve um decréscimo da utilização das redes neurais, porém, com o surgimento do *Deep Learning*, estas técnicas voltaram em força, sendo, atualmente, vistas como a tecnologia mais avançada para a descoberta de padrões em dados.

As redes neurais profundas (DNNs) tentam imitar o funcionamento do cérebro humano. Em geral, são compostas por uma camada de entrada (os inputs), uma camada de saída (os outputs) e várias camadas ocultas. A presença de camadas ocultas é o que torna uma rede neural profunda. Cada camada executa tipos específicos de classificação e ordenação em um processo chamado "hierarquia de recursos". As redes neurais profundas são instrumentos muito poderosos, mas computacionalmente exigentes. Os algoritmos desenvolvidos com base na aprendizagem profunda são usados para problemas complexos de estimação e classificação. Embora as suas origens remontem aos anos 1970, só recentemente se tornaram viáveis em grande escala devido aos avanços em *hardware* (CPUs, GPUs, TPUs), algoritmos numéricos e implementações de *software* relacionadas. Ao aplicar DNNs, é necessário um grande número de etapas repetidas, portanto um esforço computacional significativo, para ajustar os parâmetros e os híper-parâmetros que entram na arquitetura da rede.

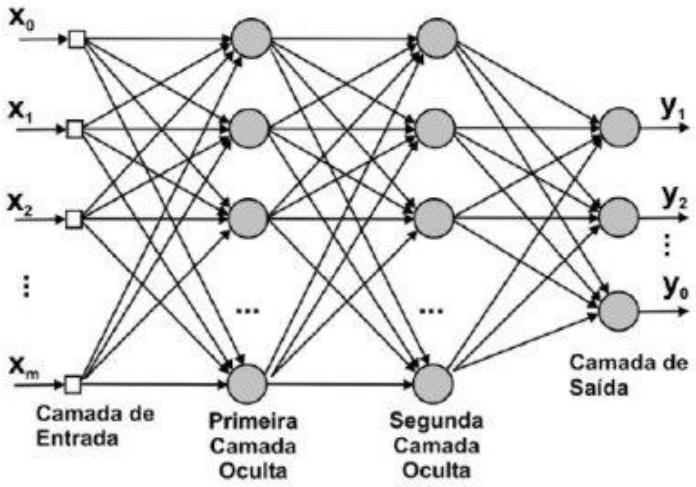

Figura 8. Rede Neural Profunda

A figura acima, ilustra a composição de uma rede com m inputs, 2 camadas ocultas com vários neurónios e uma camada de saída com 3 nodos de output. É uma rede densa, pois cada neurónio é ligado a todos os neurónios da camada seguinte.

Learning rule (Regra de aprendizagem)

O poder dos modelos de redes neurais depende em grande parte da maneira como os pesos são ajustados ao longo do tempo. Geralmente, os pesos da RNA precisam de ser ajustados usando algum algoritmo de aprendizado para que a RNA possa aproximar a função objetivo com uma precisão suficiente. Uma regra de aprendizagem é definida como um método que modifica os pesos e *bias* de uma rede. Esse método também é conhecido como algoritmo de treino. O raciocínio por trás do treino é que os pesos são atualizados de maneira a facilitar a aprendizagem dos padrões inerentes aos dados. Os dados são divididos em dois conjuntos, um conjunto de treino e um conjunto de testes. O conjunto de treino serve para estimar pesos no modelo. Portanto, o processo de aprendizagem é uma fase crucial na modelagem de redes neurais artificiais. Normalmente, o conjunto de testes consiste em 5% a 30% do conjunto total de dados.

A aprendizagem ocorre quando a rede neural atinge uma solução generalizada para uma classe de problemas. Denomina-se algoritmo de aprendizagem a um conjunto de regras bem definidas para a solução de um problema de aprendizagem. Existem muitos tipos de algoritmos de aprendizagem específicos para determinados modelos de redes neurais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados. O *backpropagation* é o algoritmo típico para o treino de redes Multi-Camadas.

Algoritmo *BackPropagation*

"Não é errando que se aprende?"

Neste trabalho será utilizado o algoritmo *backpropagation*, um dos mais potentes algoritmos relacionados com RNA, para estimar os parâmetros e pesos durante o processo de treino.

O algoritmo *backpropagation* (retropropagação) é: com base no cálculo do erro ocorrido na camada de saída da rede neural, recalcular o valor dos pesos do vetor *w* da última camada de neurónios e assim proceder para as camadas anteriores, de trás para a frente, ou seja, atualizar todos os pesos *w* das camadas a partir da última até atingir a camada de entrada da rede, realizando assim, a retropropagação do erro obtido pela rede. Na tentativa de minimizar o valor da função de erro, calculam-se os valores dos gradientes para cada peso da rede e como queremos um decréscimo na função erro, basta tomarmos o sentido contrário do gradiente. Os pesos ($W_1, W_2, \ldots, W_n$) são então modificados para minimizar o erro quadrático médio entre a previsão e o valor real. Este erro é utilizado para atualizar os pesos. Este processo é repetido até que o erro total da rede seja minimizado. Assim, a fórmula geral de atualização dos pesos na iteração fica:

$$w \leftarrow w - \eta \frac{\partial E}{\partial w} \tag{5}$$

O valor do peso na iteração atual será o valor do peso na iteração anterior, corrigido de valor proporcional ao gradiente. O sinal negativo indica que estamos a ir na direção contrária à do gradiente. O parâmetro η representa a taxa de aprendizagem da rede neural. O cálculo da expressão $\partial E/\partial w$, consiste em calcular as derivadas parciais da função de erro *E* em relação a cada peso do vetor *w*. Para uma melhor compreensão, vamos considerar a figura a seguir, que ilustra uma rede MLP com duas camadas. Uma conexão entre um neurónio *j* e um neurónio *i* da camada seguinte possui peso *w[j,i]*. Os números sobrescritos, entre parênteses, indicam o número da camada à qual a variável em questão pertence.

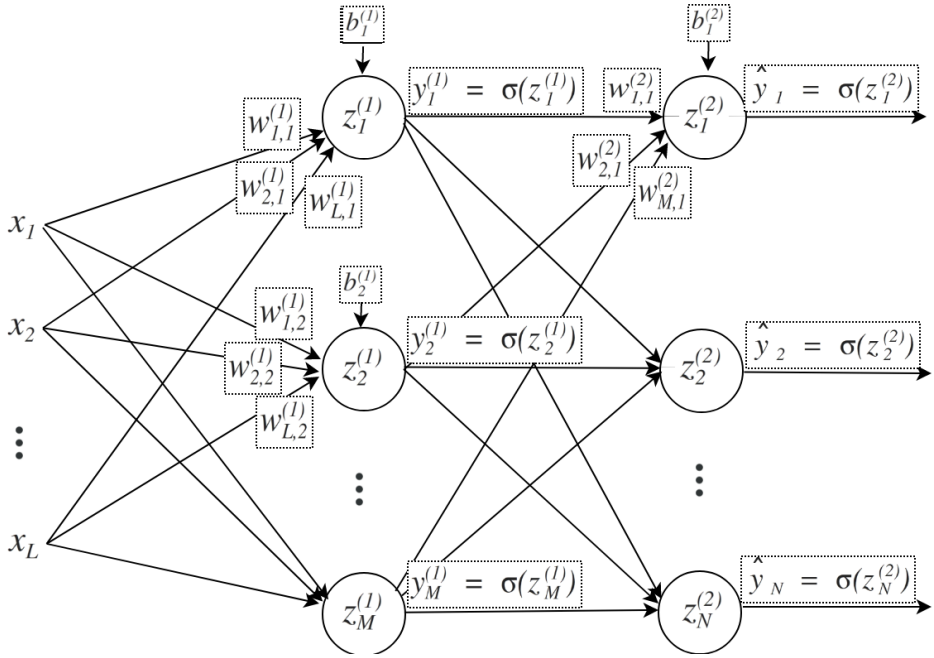

Figura 9. Rede MLP para ilustrar a explicação do *backpropagation*

Seja y a saída esperada e $\hat{y}$ a saída obtida pela rede, a função de erro é definida da seguinte forma:

$$E(y, \hat{y}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \quad (6)$$

Ou seja, a soma dos quadrados dos resíduos. A função erro também denota-se por *loss* (perda) ou custo.

Aprendizagem supervisionada

Existem basicamente três técnicas de aprendizagem: supervisionada, não-supervisionada e semi-supervisionada. Neste trabalho, vai ser usada a aprendizagem supervisionada.

Na aprendizagem de máquina supervisionada é fornecido ao algoritmo um conjunto de inputs para treino. Os inputs já contêm o que o algoritmo de *machine learning* deve aprender. Assim, após o treino da rede com os dados pré-definidos, o programa é capaz de tomar decisões precisas quando recebe novos dados.

Método de otimização L-BFGS

L-BFGS (*Limited- memory* Broyden-Fletcher-Goldfarb-Shanno) é um método de otimização numérica com capacidade de aproximar a matriz hessiana de uma função que não admite derivadas parciais. A vantagem do L-BFGS é que requer apenas a manutenção de *m* gradientes mais recentes, em que *m é* geralmente de 10 a 20, que é um requisito de armazenamento muito menor do que *n* * (*n* + 1) / 2 elementos necessários para armazenar a totalidade de uma estimativa hessiana, como é exigido no método clássico BFGS, em que *n* é a dimensão do problema. O L-BFGS é usado em vez do BFGS para problemas muito grandes (quando n é muito grande). A função de otimização L-BFGS ajuda a encontrar os pesos que produzirão uma perda menor na próxima iteração.

MLPClassifier

MLPClassifier é um algoritmo de classificação que significa *Multi-layer Perceptron (MLP) Classifier* (classificador Perceptron de várias camadas). Ao contrário de outros algoritmos de classificação, como vetores de suporte ou *Naive Bayes Classifier*, o *MLPClassifier* conta com uma rede neural subjacente para executar a tarefa de classificação. A principal vantagem do Multi-layer Perceptron é a sua capacidade de aprender e quantificar modelos não lineares. Contudo, também existem desvantagens, o MLP com camadas ocultas tem uma função de perda não convexa onde existe mais de um mínimo local e, portanto, diferentes inicializações de peso aleatório podem levar a uma precisão de validação diferente. Além disso, o MLP requer o ajuste de vários parâmetros.

DNNs com *scikit-learn*

A *scikit-learn* é uma biblioteca de Machine Learning de código aberto para a linguagem de programação Python. O *scikit-learn* fornece a mesma API para sua classe de algoritmo *MLPClassifier*, que é um modelo DNN, como para os outros algoritmos ML usados anteriormente. Com poucas camadas ocultas (por exemplo duas), alcança um resultado perfeito nos dados de teste (as camadas ocultas é o que torna a aprendizagem profunda diferente da aprendizagem simples).

3. Resultados e análise

O objetivo deste estudo é utilizar estratégias de *trading* para prever a direção do movimento do índice bolsista português, PSI 20, analisá-las quanto à sua eficácia e efeitos.

A série considerada tem dados diários (5 dias por semana) compreendidos no período temporal 04/01/2010 a 03/07/2019, tendo sido retirados da base de dados DataStream. Os códigos de Python utilizados nos três métodos foram retirados de: Hilpisch, Y. (2019)***. Python for Finance: Mastering Data-Driven Finance***. Sebastopol: O'Reilly.

A partir de uma análise exploratória da série em estudo, podemos concluir que não tem uma tendência global e não pertence a uma distribuição normal.

Vejamos agora os resultados da aplicação dos três algoritmos de *trading* para o índice PSI20.

3.1. *Simple Moving Average*

A estratégia que vai ser implementada é com base na tendência a dois meses (42 dias de *trading*) e um ano (252 dias de *trading*). Como a utilização das bibliotecas do Python (Pandas), é possível visualizar as três séries temporais numa única figura. Intitulamos o *moving average* a 42 dias como SMA1 e o *moving average* a 252 dias como SMA2. Facilmente se verifica que ao utilizar um *moving average* a um conjunto de dados mais pequeno, neste caso 42 dias, a série aproxima-se mais dos preços de fechos do índice, uma vez que faz alisamentos em intervalos menores.

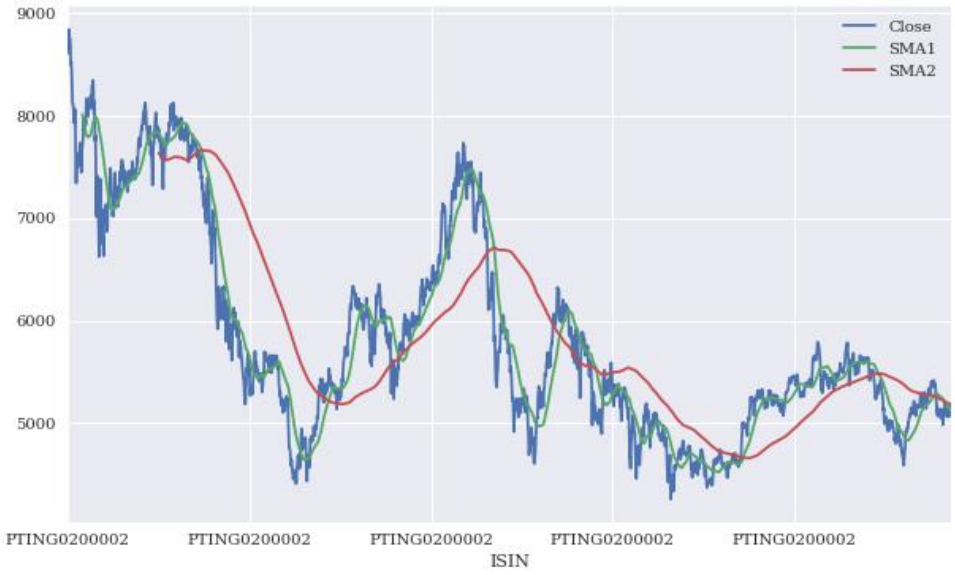

Figura 10. O índice PSI 20 e as linhas de tendência a 42 dias e 252 dias

Considera-se uma regra para gerar sinais de *trading* que diz o seguinte: se a tendência a 42 dias está pela primeira vez, acima da tendência a 252 dias então é um sinal de compra, se a tendência a 42 dias está pela primeira vez, abaixo da tendência a 252 dias então é um sinal de venda e devemos

mantê-lo nas restantes situações. Assim, temos tudo disponível para testar a estratégia de investimento com base nos sinais. Assumimos, para simplificar, que um investidor pode investir diretamente no índice. Tais transações têm inevitavelmente custos de transação, que não iremos considerar. Isto é justificável uma vez que não planeamos fazer transações com muita frequência. Um investidor que assuma uma posição curta em títulos tem uma visão negativa do mercado, enquanto que se diz que alguém que assume uma posição longa tem uma visão otimista. O investidor pode assumir uma posição curta ou longa no mercado ou então manter em dinheiro a sua riqueza, mas isto não tem nenhum interesse. Esta estratégia simplificada permite que trabalhemos apenas com os retornos do mercado. O investidor faz o retorno do mercado positivo quando é na posição longa (1) e faz o retorno do mercado negativo quando a sua posição é curta (-1). Não existe retorno (0), quando o investidor lida com liquidez da sua riqueza.

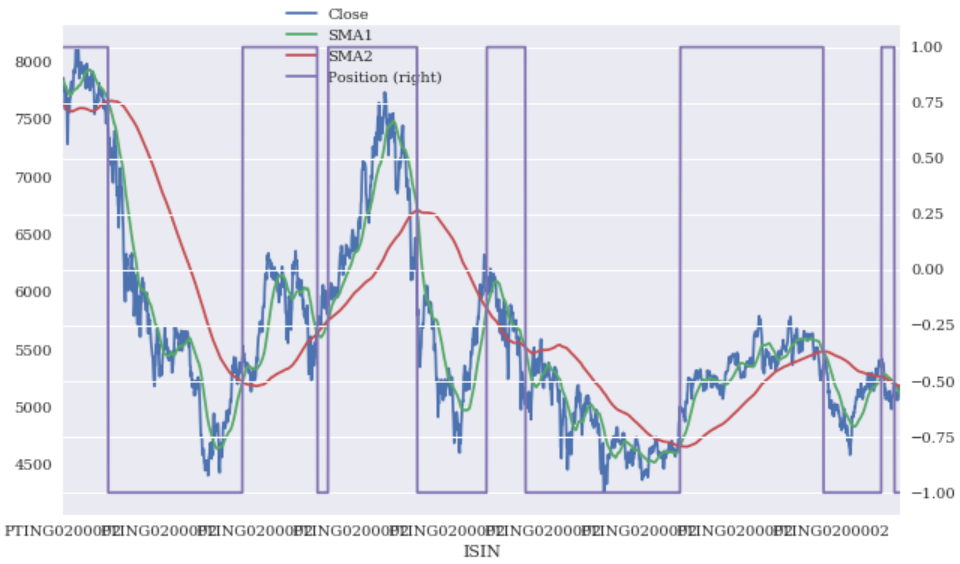

Figura 11. Sinais de compra e venda ao longo do tempo

Agora é simples obter os retornos da estratégia de negociação baseada em tendências, basta multiplicar a coluna *Position*, alterada por um dia, pelas colunas de retorno. Como podemos visualizar na figura abaixo.

Figura 12. A estratégia e os retornos do PSI 20 com base no método de *simple moving average*

Esta figura compara os retornos cumulativos e contínuos do índice com os retornos cumulativos e contínuos de nossa estratégia. É possível visualizar que a estratégia compensa bem pois o investidor é capaz de obter um retorno muito maior sobre o período relevante do que um investimento a longo prazo proporcionaria. Esta figura mostra-nos que, especialmente durante as desacelerações do mercado, o *shorting* de mercado produz retornos bastante altos. O tempo de execução desta estratégia é 4 segundos.

3.2. *Frequency Approach*

Este método mostra a frequência de movimentos possíveis condicionais nas combinações de valor de recurso. Depois, destaca-se o valor da frequência mais alta por combinação de valor do recurso.

Tabela 1. Contagem de frequências com base nas combinações

	direction	-1	1
lag_1_bin	lag_2_bin		
0	0	316	321
	1	321	239
1	0	268	291
	1	292	381

Neste método foram utilizados dois *lags*, sendo possíveis as seguintes combinações: (0,0),(0,1),(1,0) e (1,1). Através da análise da tabela com os dados das frequências, observam-se três combinações de valores que sugerem um movimento ascendente, enquanto apenas uma permite que um movimento descendente pareça mais provável. O desempenho desta estratégia pode ser visto na figura abaixo:

Figura 13. A estratégia e os retornos do PSI 20 com base no método de *frequency approach*

Em comparação com o *moving average*, esta estratégia apresenta um melhor resultado, uma vez que inicialmente a estratégia está próxima dos retornos, mas com a evolução temporária vão dispersando-se, tornando-se menos eficaz. Esta estratégia demora 15 segundos até ser apresentada.

3.3. Rede Neural Artificial

Nesta estratégia iremos utilizar uma rede neural artificial com *backprogagation* e aprendizagem supervisionada profunda. Após correr o programa várias vezes, a arquitetura da rede neural escolhida é a seguinte:

MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=[100, 100], learning_rate='constant', learning_rate_init=0.001, max_iter=500, momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False)

Apesar da complexidade deste método em relação aos dois anteriores, a rede leva 19 segundos para apresentar a sua estratégia sobre o conjunto de teste (10% da amostra total) e como se pode observar na figura abaixo, a estratégia é bastante próxima dos retornos, ultrapassando o tempo e a performance dos outros algoritmos.

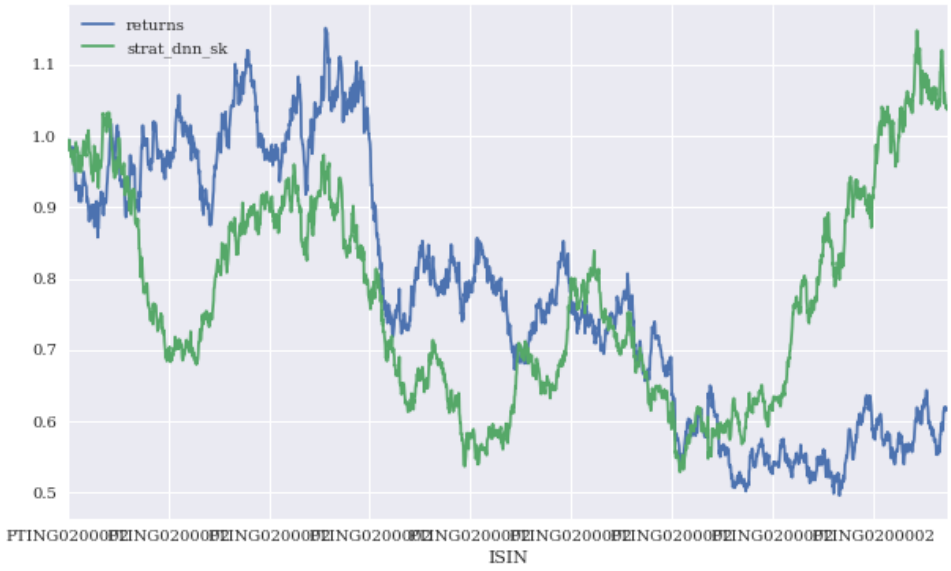

Figura 14. A estratégia e os retornos do PSI 20 com base no método da rede neural com *backpropagation* e aprendizagem supervisionada profunda sobre conjunto teste de 10% da amostra total

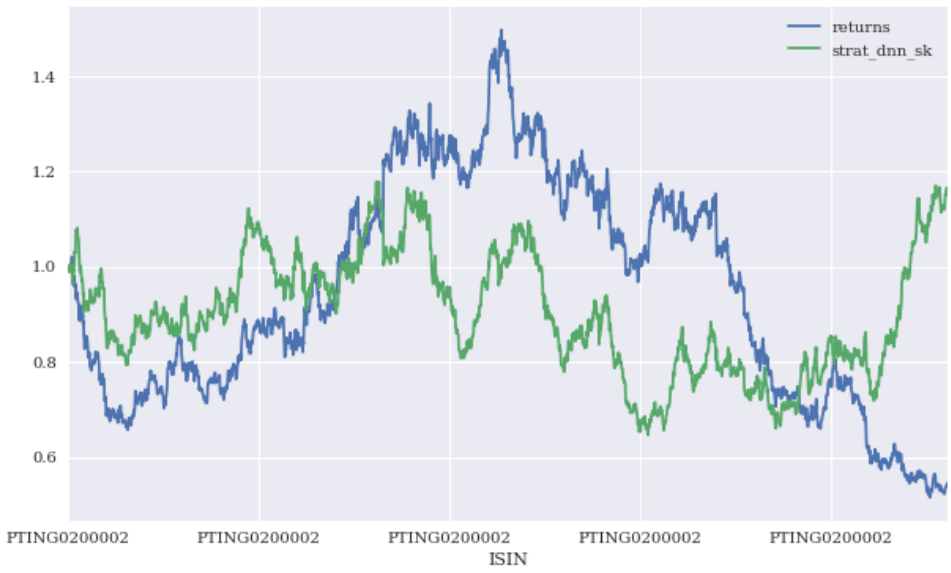

Figura 15. A estratégia e os retornos do PSI 20 com base no método da rede neural com *backpropagation* e aprendizagem supervisionada profunda sobre conjunto teste de 5% da amostra total

Alterando o tamanho do conjunto de teste (apenas 5% neste caso), e mantendo os valores dos outros parâmetros, observa-se um incremento na fidelização do algoritmo. Tanto as diferenças entre a estratégia e o retorno como o tempo de obtenção do output decrescem de forma visível.

Conclusão

A crescente evolução da tecnologia fez com que o *algorithmic trading* se tornasse uma ferramenta essencial para que os investidores pudessem otimizar as suas estratégias e, consequentemente, aumentar os seus lucros.

Neste estudo testamos três estratégias de *trading* com o objetivo de prever os movimentos ascendentes ou descendentes dos preços do PSI 20.

Através dos resultados obtidos na estratégia do *simple moving average,* verificamos que este método é mais vantajoso quando a série temporal segue uma tendência específica uma vez que este método não toma em consideração os picos da série. Se fosse utilizado o método *weight moving average*, provavelmente os resultados seriam um pouco melhores porque poderíamos dar um peso menor às observações referentes às datas onde houve picos (*outliers*). Com base na série do PSI 20, verificamos que a estratégia está muito afastada dos retornos, pelo que se conclui que este método não é o mais adequado para os nossos dados.

O método de *frequency approach*, é um método que se adequa melhor aos dados históricos do PSI 20 do que o método anterior, mas, ainda assim, é uma estratégia que só funciona bem no início, em que a estratégia está muito próxima dos retornos e, depois, rapidamente se dispersam.

No método das redes neurais profundas a estratégia adotada aproxima-se dos retornos, sendo, portanto, uma boa estratégia para utilizar na série temporal em estudo.

As estratégias utilizadas neste trabalho surgiram em períodos distintos. E, portanto, era espectável que a estratégia mais recente e complexa, a rede neural artificial com *backpropagation* e aprendizagem supervisionada profunda, fosse aquela que retornasse melhores resultados.

Conclui-se que o *algorithmic trading* tem tido um papel fundamental nos mercados financeiros, na medida em que melhorou a eficácia do processo de tomada de decisões por parte dos investidores, aumento de liquidez, eliminação ou diminuição das ineficiências de mercado, custos reduzidos.

É de notar que todo o conteúdo desta dissertação nunca foi abordado nas cadeiras do mestrado, daí poder acusar alguma incompletude.

Referências Bibliográficas

Alpaydin, E. (2014**) Introduction to machine learning**. London: MIT press

Amal. (2019). ***A Beginner's Guide to Scikit- Learn's MLPClassifier***. Obtido de: https://www.analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/

Capgemini DE. (2018). ***Why algorithmic trading constitutes a crucial success factor for energy market participants***. Obtido de: https://www.capgemini.com/de-de/2018/07/algorithmic-trading-part-1/#

Chan, E. (2017). ***Machine trading: Deploying computer algorithms to conquer the markets***. Hoboken: Wiley

Chan, E. (2009). ***Quantitative Trading - How to Build Your Own Algorithmic Trading Business***. Hoboken: Wiley

Chun, C. (2012). ***Cross-border Transactions of Intermediated Securities***. Berlim: Springer-Verlag Berlin Heidelberg

Computershare. (2013). ***21st Century Stock Ownership: Eliminating Paper Certificates and Expanding Direct Registration***. Obtido de: http://www.computershare.com/News/21st_Century_Stock_Ownership.pdf

Coyle, J, Langley, C, Gibson, B, Novack, R. & Bardi, E. (2009). ***Supply Chain Management: A logistics Perspective***, 8th edition. USA: Cengage Learning

Definition of 'Algorithm Trading'. Obtido de: https://economictimes.indiatimes.com/definition/algorithm-trading. Consultado em: maio de 2019

DINO. (2018). ***Deep Learning: o cérebro por trás da Inteligência Artificial***. Obtido de: https://exame.abril.com.br/negocios/dino/deep-learning-o-cerebro-por-tras-da-inteligencia-artificial/

Dixon, M, Klabjan, D & Bang, J. (2017). ***Classification-based financial markets prediction using deep neural networks***. USA

Evolution Of The Marketplace: From Open Outcry To Electronic Trading. Obtido de: https://www.fxcm.com/uk/insights/evolution-of-the-marketplace-from-open-outcry-to-electronic-trading/. Consultado em: maio de 2019

Facure, M. (2017). ***Uma apresentação teórica e intuitiva às redes neurais artificiais, Introdução às Redes Neurais Artificiais***. Obtido de: https://matheusfacure.github.io/2017/03/05/ann-intro/

Gilberto, F. (2015). ***Negociação Algorítmica de Alta Frequência – negócios à velocidade da luz***. Portp: Vida Económica – Editorial, SA

Gill, N. (2019). ***Artificial Neural Networks Applications and Algorithms***. Obtido de: https://www.xenonstack.com/blog/artificial-neural-network-applications/

Gsell, M. (2008). ***Assessing the impact of algorithmic trading on markets: A simulation approach***. CFS Working Paper Series 2008/49

Harris, L. (2003). ***Trading and Exchanges: Market Microstructure for Practitioners***. New York: Oxford University Press

Hecht, R. (1989). ***Theory of the backpropagation neural network***. In: Neural Networks. IJCNN., International Joint Conference on. IEEE, 1989. p. 593-605.

Hilpisch, Y. (2014). ***Python for Finance: Analyze Big Financial Data***. Sebastopol: O'Reilly

Hilpisch, Y. (2019)***. Python for Finance: Mastering Data-Driven Finance***. Sebastopol: O'Reilly

Indicadores Técnicos utilizados na Bolsa de Valores. Obtido de: https://www.estrategia-bolsa.pt/indicadores-tecnicos-bolsa-valores.html. Consultado em: junho de 2019

Kant, R. (2019). ***Why algorithmic trading is dangerous***. Obtido de: https://www.asiatimes.com/2019/05/opinion/why-algorithmic-trading-is-dangerous/

Kim, K. (2007). ***Electronic and Algorithmic Trading Technology: The Complete Guide***. Burlington: Academic Press

Kissell, R. (2014). ***The Science of Algorithmic Trading and Portfolio***. San Diego: Academic Press

Leshik, E. & Cralle, J. (2011). ***An Introduction of Algorithmic Trading: Basic to Advanced Strategies***. United Kingdom: Wiley

Machado, J, Oliveira, R & Pereira, A (2018). ***Proposal and Implementation of Machine Learning and Deep Learning Models for Stock Markets***. Belo Horizonte

Mahadevan, B. (2010). ***Operations Management: Theory and Practise***. Dorling Kindersley: Pearson

Matos, D. (2015). ***Conceitos Fundamentais de Machine Learning.*** Obtido de: http://www.cienciaedados.com/conceitos-fundamentais-de-machine-learning/

Média Movel (Moving Average Indicator). Obtido de: https://www.ifcmarkets.pt/ntx-indicators/moving-average. Consultado em: junho de 2019

Mislinski, J. (2019). ***Moving Averages: August Month-End Update***, Obtido de: https://www.advisorperspectives.com/dshort/updates/2019/08/30/moving-averages-august-month-end-update

Moolayil, J. (2019***). Learn Keras for Deep Neural Networks***. Vanconver: Apress

Nascimento, A. (2017). ***"A desmaterialização no mercado financeiro: evolução e prospetiva"***. Instituto Superior de Economia e Gestão da Universidade de Lisboa. Dissertação de Mestrado

Neural network models (supervised). Obtido de: https://scikit-learn.org/stable/modules/neural_networks_supervised.html. Consultado em: junho de 2019

Nicolas, P. (2015). ***Scala for Machine Learning.*** Birmingham: Packt Publishing

Oliveira, T. (2017). ***"High Frequency Trading" – O Novo Paradigma da Negociação Automatizada***. Faculdade de Direito de Lisboa da Universidade Católica Portuguesa. Dissertação de Mestrado

Oliveira, W. (2011). ***A Aplicação de Redes Neuronais na Detecção da Influênciado High Frequency Trading na Negociaçãode Acções (Caso Português)***. Instituto Superior de Estatística e Gestão da Informação da Universidade Nova de Lisboa. Dissertação de Mestrado

Premkumar, D. (2019). ***What is Algorithmic Trading (Algo Trading)? And how it works?***. Obtido de: https://tradebrains.in/algorithmic-trading/

PSI-20. (2016). Obtido de: https://www.economias.pt/psi-20/

Reilly, F, & Brown, K. (2011). ***New Trading Systems***. F. K. Reilly & K. C. Brown (Eds.), Investment Analysis & Portfolio Management: 117-118.USA: South-Western Cengage Learning.

Rodrigues, J. (2012). ***Algoritmos "assaltam" mercados de commodities depois das bolsas***, Jornal Expresso

Sharma, R. (2019). ***Podcast | Digging Deeper: What is algorithmic trading and how can you get into it?***. Obtido de: https://www.moneycontrol.com/news/business/markets/podcast-digging-deeper-what-is-algorithmic-trading-and-how-can-you-get-into-it-3377101.html

Shift Markets. (2019). ***Advantages of Algorithmic Trading***. Obtido de: https://www.nasdaq.com/articles/advantages-algorithmic-trading-2019-06-07

Silva, J.E. (2015). Metodologia, Tese de Mestrado em: ***Modelagem e Aplicação de Técnicas de Aprendizado de Máquina para Negociação em Alta Frequência em Bolsa de Valores.*** Universidade Federal de Minas Gerais- Departamento de Ciência da Computação, Belo Horizonte, p.13.

Silva, P. (2012). ***Optimization of Techical Trading Rules In Forex Market Using Genetic Algorithm,*** ISCTE Business School. Dissertação de Mestrado

Simões, V. (2011). O Impacto da Especulação nos Mercados Financeiros. Universidade de Aveiro. Dissertação de Mestrado

Sinais de Negociação no Forex. (2013). Obtido de: https://expresso.pt/gictreze/gictrezemontepio/sinais-de-negociacao-no-forex=f844482

Sousa, I. (2016). ***Negociação Algorítmica de Alta Frequência- Questões Jurídicas e Económicas.*** Faculdade de Direito da Universidade de Coimbra. Dissertação de Mestrado

Stone, M. (2017). ***How does the L-BFGS work?***. Obtido de: https://stats.stackexchange.com/questions/284712/how-does-the-l-bfgs-work

The TowerGroup, s.v. ***"Algorithmic Trading" Glossary of Terms.*** Obtido de: http://www.towergroup.com/research/content/glossary.jsp?page=1&glossaryId=382

Toth, M. (2013). ***How was trading done in stock market when there were no computers?.*** Obtido de: https://www.quora.com/How-was-trading-done-in-stock-market-when-there-were-no-computers

Trading Economics (2019). Portugal Stock Market (PSI20). Obtido de: https://tradingeconomics.com/portugal/stock-market

Vaz, A. (2018). ***Introdução Teórica a Neural Network- Deep Learning- Parte 1***. Obtido de: https://medium.com/data-hackers/neural-network-deep-learning-parte-1-introdução-teórica-5c6dcd2e5a79

Verheggen, R. (2017). ***The Rise of Algorithmic Trading and its effects on Return Dispersion and Market Predictability***. Tilburg University. Dissertação de Mestrado

X-Trade Brokers Dom Maklerski S.A. ***Médias Móveis***. Obtido de: https://www.xtb.com/pt/aprender-a-negociar/trading-academy-medias-moveis. Consultado em: junho de 2019

Zambiasi, S. ***O Neurônio Artificial***. Obtido de: https://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio_implementacao/. Consultado em: junho de 2019

Anexos

Algoritmo para a estratégia de *Moving Average:*

```
import numpy as np
import pandas as pd
import datetime as dt
from pylab import mpl, plt

plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'
get_ipython().run_line_magic('matplotlib', 'inline')

raw = pd.read_csv('C:/Users/DEBORA/Desktop/py4fi2nd-master/source/psi20.csv',
                  index_col=0, parse_dates=True)

raw.info()

symbol = 'Close'

data = (
    pd.DataFrame(raw[symbol])
    .dropna()
)

SMA1 = 42
SMA2 = 252

data['SMA1'] = data[symbol].rolling(SMA1).mean()
data['SMA2'] = data[symbol].rolling(SMA2).mean()

data.plot(figsize=(10, 6));
plt.savefig(' C:/Users/DEBORA/Desktop/imagens')
```

```
data.dropna(inplace=True)

data['Position'] = np.where(data['SMA1'] > data['SMA2'], 1, -1)

data.tail()

ax = data.plot(secondary_y='Position', figsize=(10, 6))
ax.get_legend().set_bbox_to_anchor((0.25, 0.85));
plt.savefig(' C:/Users/DEBORA/Desktop/imagens')



# ### Vectorized Backtesting
data['Returns'] = np.log(data[symbol] / data[symbol].shift(1))

data['Strategy'] = data['Position'].shift(1) * data['Returns']

data.round(4).head()

data.dropna(inplace=True)

np.exp(data[['Returns', 'Strategy']].sum())

data[['Returns', 'Strategy']].std() * 252 ** 0.5

ax = data[['Returns', 'Strategy']].cumsum(
      ).apply(np.exp).plot(figsize=(10, 6))
data['Position'].plot(ax=ax, secondary_y='Position', style='--')
ax.get_legend().set_bbox_to_anchor((0.25, 0.85));
plt.savefig(' C:/Users/DEBORA/Desktop/imagens);
```

```
# ### Optimization
from itertools import product

sma1 = range(20, 61, 4)
sma2 = range(180, 281, 10)

results = pd.DataFrame()
for SMA1, SMA2 in product(sma1, sma2):
    data = pd.DataFrame(raw[symbol])
    data.dropna(inplace=True)
    data['Returns'] = np.log(data[symbol] / data[symbol].shift(1))
    data['SMA1'] = data[symbol].rolling(SMA1).mean()
    data['SMA2'] = data[symbol].rolling(SMA2).mean()
    data.dropna(inplace=True)
    data['Position'] = np.where(data['SMA1'] > data['SMA2'], 1, -1)
    data['Strategy'] = data['Position'].shift(1) * data['Returns']
    data.dropna(inplace=True)
    perf = np.exp(data[['Returns', 'Strategy']].sum())
    results = results.append(pd.DataFrame(
                {'SMA1': SMA1, 'SMA2': SMA2,
                 'MARKET': perf['Returns'],
                 'STRATEGY': perf['Strategy'],
                 'OUT': perf['Strategy'] - perf['Returns']},
                 index=[0]), ignore_index=True)

results.info()

results.sort_values('OUT', ascending=False).head(7)
```

Algoritmo para estratégias de *Frequency Aproach* e Rede Neural Artificiais

```
import numpy as np
```

```
import pandas as pd
import datetime as dt
from pylab import mpl, plt
import warnings

warnings.simplefilter('ignore')
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'
np.random.seed(1000)
get_ipython().run_line_magic('matplotlib', 'inline')

# Linear OLS Regression

# The Data

raw = pd.read_csv('C:/Users/DEBORA/Desktop/py4fi2nd-master/source/psi20.csv',
                  index_col=0, parse_dates=True).dropna()

raw.columns

symbol = 'Close'

data = pd.DataFrame(raw[symbol])

data['returns'] = np.log(data / data.shift(1))

data.dropna(inplace=True)

data['direction'] = np.sign(data['returns']).astype(int)

data.head()
```

```
data['returns'].hist(bins=35, figsize=(10, 6));
plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig1')
lags = 2

def create_lags(data):
    global cols
    cols = []
    for lag in range(1, lags + 1):
        col = 'lag_{}'.format(lag)
        data[col] = data['returns'].shift(lag)
        cols.append(col)

create_lags(data)

data.head()

data.dropna(inplace=True)

data.plot.scatter(x='lag_1', y='lag_2', c='returns',
                  cmap='coolwarm', figsize=(10, 6), colorbar=True)
plt.axvline(0, c='r', ls='--')
plt.axhline(0, c='r', ls='--');
plt.savefig(':/Users/DEBORA/Desktop/imagens/fig2');

# Regression
from sklearn.linear_model import LinearRegression

model = LinearRegression()

data['pos_ols_1'] = model.fit(data[cols], data['returns']).predict(data[cols])

data['pos_ols_2'] = model.fit(data[cols], data['direction']).predict(data[cols])
```

```
data[['pos_ols_1', 'pos_ols_2']].head()

data[['pos_ols_1', 'pos_ols_2']] = np.where(
        data[['pos_ols_1', 'pos_ols_2']] > 0, 1, -1)

data['pos_ols_1'].value_counts()

data['pos_ols_2'].value_counts()

(data['pos_ols_1'].diff() != 0).sum()

(data['pos_ols_2'].diff() != 0).sum()

data['strat_ols_1'] = data['pos_ols_1'] * data['returns']

data['strat_ols_2'] = data['pos_ols_2'] * data['returns']

data[['returns', 'strat_ols_1', 'strat_ols_2']].sum().apply(np.exp)

(data['direction'] == data['pos_ols_1']).value_counts()

(data['direction'] == data['pos_ols_2']).value_counts()

data[['returns', 'strat_ols_1', 'strat_ols_2']].cumsum(
      ).apply(np.exp).plot(figsize=(10, 6));
plt.savefig(':/Users/DEBORA/Desktop/imagens/fig3');

# Frequency Approach

def create_bins(data, bins=[0]):
```

```
    global cols_bin
    cols_bin = []
    for col in cols:
        col_bin = col + '_bin'
        data[col_bin] = np.digitize(data[col], bins=bins)
        cols_bin.append(col_bin)

create_bins(data)

data[cols_bin + ['direction']].head()

grouped = data.groupby(cols_bin + ['direction'])
grouped.size()

res = grouped['direction'].size().unstack(fill_value=0)

def highlight_max(s):
    is_max = s == s.max()
    return ['background-color: yellow' if v else '' for v in is_max]

res.style.apply(highlight_max, axis=1)

data['pos_freq'] = np.where(data[cols_bin].sum(axis=1) == 2, -1, 1)

(data['direction'] == data['pos_freq']).value_counts()

data['strat_freq'] = data['pos_freq'] * data['returns']

data[['returns', 'strat_freq']].sum().apply(np.exp)

data[['returns', 'strat_freq']].cumsum().apply(np.exp).plot(figsize=(10, 6));
plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig4');
```

```
# Classification Algorithms

from sklearn import linear_model
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

C = 1

models = {
    'log_reg': linear_model.LogisticRegression(C=C),
    'gauss_nb': GaussianNB(),
    'svm': SVC(C=C)
}

def fit_models(data):
    mfit = {model: models[model].fit(data[cols_bin], data['direction'])
            for model in models.keys()}

fit_models(data)

def derive_positions(data):
    for model in models.keys():
        data['pos_' + model] = models[model].predict(data[cols_bin])
derive_positions(data)

def evaluate_strats(data):
    global sel
    sel = []
    for model in models.keys():
        col = 'strat_' + model
```

```
        data[col] = data['pos_' + model] * data['returns']
        sel.append(col)
    sel.insert(0, 'returns')

evaluate_strats(data)

sel.insert(1, 'strat_freq')

data[sel].sum().apply(np.exp)

data[sel].cumsum().apply(np.exp).plot(figsize=(10, 6));
 plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig5')

data = pd.DataFrame(raw[symbol])

data['returns'] = np.log(data / data.shift(1))

data['direction'] = np.sign(data['returns'])

lags = 5
create_lags(data)
data.dropna(inplace=True)

create_bins(data)
cols_bin

data[cols_bin].head()

data.dropna(inplace=True)

fit_models(data)
```

```
derive_positions(data)

evaluate_strats(data)

data[sel].sum().apply(np.exp)

data[sel].cumsum().apply(np.exp).plot(figsize=(10, 6));
plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig6’');

mu = data['returns'].mean()
v = data['returns'].std()

bins = [mu - v, mu, mu + v]
bins

create_bins(data, bins)

data[cols_bin].head()

fit_models(data)

derive_positions(data)

evaluate_strats(data)

data[sel].sum().apply(np.exp)

data[sel].cumsum().apply(np.exp).plot(figsize=(10, 6));
plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig7')

# Sequential Train-Test Split
```

```
split = int(len(data) * 0.5)

train = data.iloc[:split].copy()

fit_models(train)

test = data.iloc[split:].copy()

derive_positions(test)

evaluate_strats(test)

test[sel].sum().apply(np.exp)

test[sel].cumsum().apply(np.exp).plot(figsize=(10, 6));
plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig8’);

# Randomized Train-Test Split

from sklearn.model_selection import train_test_split

train, test = train_test_split(data, test_size=0.10,
                               shuffle=True, random_state=100)

train = train.copy().sort_index()

train[cols_bin].head()

test = test.copy().sort_index()

fit_models(train)
```

```
derive_positions(test)

evaluate_strats(test)

test[sel].sum().apply(np.exp)

test[sel].cumsum().apply(np.exp).plot(figsize=(10, 6));
plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig9');

# Deep Neural Network

## DNN with scikit-learn
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(solver='lbfgs', alpha=1e-5,
                      hidden_layer_sizes=2 * [100], random_state=1)

get_ipython().run_line_magic('time', "model.fit(data[cols_bin], data['direction'])")

data['pos_dnn_sk'] = model.predict(data[cols_bin])

data['strat_dnn_sk'] = data['pos_dnn_sk'] * data['returns']

data[['returns', 'strat_dnn_sk']].sum().apply(np.exp)

data[['returns', 'strat_dnn_sk']].cumsum().apply(np.exp).plot(figsize=(10, 6));
# plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig10');

train, test = train_test_split(data, test_size=0.10, random_state=100)

train = train.copy().sort_index()
```

```
test = test.copy().sort_index()

model = MLPClassifier(solver='lbfgs', alpha=1e-5, max_iter=500,
                      hidden_layer_sizes= 2 * [100], random_state=1)

get_ipython().run_line_magic('time', "model.fit(train[cols_bin], train['direction'])")

test['pos_dnn_sk'] = model.predict(test[cols_bin])

test['strat_dnn_sk'] = test['pos_dnn_sk'] * test['returns']

test[['returns', 'strat_dnn_sk']].sum().apply(np.exp)

test[['returns', 'strat_dnn_sk']].cumsum().apply(np.exp).plot(figsize=(10, 6));
plt.savefig('C:/Users/DEBORA/Desktop/imagens/fig11');
```